AMENDMENTS TO THE CLAIMS

Please amend claims 1-28, as shown below.

1. (Currently amended) A method for generating a checkpoint for a virtual machine (VM) in a virtual computer system, the VM using a parent disk file and a set of VM memory, the method comprising:

creating a copy-on-write (COW) disk file pointing to the parent disk file in use by the VM;

stopping the VM, and while the VM is stopped:

marking the memory of the VM copy-on-write, the VM memory constituting original VM memory;

saving substantially all of the device state of the VM, including an instruction pointer, a plurality of registers and settings for one or more virtual devices, to memory; and

switching the VM to ~~use~~ read from and write to the COW disk file instead of the parent disk file;

resuming operation of the VM, so that the VM reads from and writes to the COW disk file and so that the VM reads from, writes to, and executes code from the VM memory;

handling disk COW faults to the COW disk file;

handling memory COW faults to the original VM memory to generate copies of the original VM memory for read, ~~and~~ write, and execution use by the VM;

saving the device state from memory to a checkpoint data store; and

saving the original VM memory to the checkpoint data store,

wherein the VM memory is a subset of a physical memory in the virtual computer system, the VM memory being a portion of the physical memory that is allocated to the VM.

2. (Original) The method of claim 1, wherein the VM is still running when the COW disk file is created.

3. (Original) The method of claim 1, further comprising copying the parent disk file after any pending disk writes complete, and using the copy of the parent disk file for the checkpoint.

4. (Original) The method of claim 3, further comprising committing the COW disk file into the original parent disk file.

5. (Currently amended) The method of claim ~~3~~ 4, wherein the step of committing the COW disk file into the original parent disk file comprises creating one or more new COW disk files for use by the VM while the COW disk file previously used by the VM is being committed.

6. (Original) The method of claim 3, wherein the copy of the parent disk file is indicated for use for the checkpoint by adding a disk file pointer to the checkpoint file.

7. (Original) The method of claim 1, wherein the steps of creating the COW disk file and handling disk COW faults are performed by a data storage device that is external to the virtual computer system.

8. (Original) The method of claim 1, wherein the step of saving the original VM memory to the checkpoint data store is delayed until all pending disk reads complete.

9. (Original) The method of claim 1, wherein the checkpoint data store comprises raw data stored in a data storage medium.

10. (Original) The method of claim 1, wherein the checkpoint data store comprises a file stored in a data storage medium.

11. (Original) The method of claim 10, wherein the data storage medium comprises a disk drive.

12. (Original) The method of claim 10, wherein the data storage medium comprises memory.

13. (Original) The method of claim 12, wherein the memory comprises high-speed random access memory.

14. (Original) The method of claim 12, wherein the memory comprises flash memory.

15. (Original) The method of claim 1, further comprising, after the operation of the VM is resumed, forcing memory COW faults on original VM memory that is affected by any new disk reads, prior to issuing the new disk reads.

16. (Original) The method of claim 1, wherein, if there is a pending read followed by a pending write to the same disk block, the resumption of the operation of the VM is delayed until all pending disk operations complete.

17. (Original) The method of claim 16, further comprising, if the resumption of the operation of the VM is not delayed until all pending disk operations complete, reissuing any disk reads that affect any VM memory for which a COW fault has occurred.

18. (Original) The method of claim 1, wherein, for the step of handling disk COW faults to the COW disk file, if there is a pending disk write to the same COW block group for which the disk COW fault has occurred, the handling of the disk COW fault and the write that caused the disk COW fault are delayed until the pending disk write completes.

19. (Currently amended) A method for generating a checkpoint for a virtual machine (VM) in a virtual computer system, the VM using a virtual disk and a set of VM memory, the method comprising:

maintaining, in an unmodified state, the contents of the virtual disk at the time for which the checkpoint is generated, while allowing the VM to continue ~~using~~ reading from and writing to the virtual disk;

saving substantially all of the device state of the VM, including an instruction pointer, a plurality of registers and settings for one or more virtual devices, at the time for which the checkpoint is generated, to a checkpoint data store; and

saving the set of VM memory, at the time for which the checkpoint is generated, to the checkpoint data store by performing the following steps:

marking the set of VM memory as copy-on-write (COW), the set of VM memory constituting original VM memory;

allowing the VM to continue ~~using~~ reading from, writing to, and executing code from the VM memory;

responding to memory COW faults related to the VM memory by generating copies of the original VM memory for read, ~~and~~ write, and execution use by the VM; and

saving the original VM memory to the checkpoint data store,

wherein the VM memory is a subset of a physical memory in the virtual computer system, the VM memory being a portion of the physical memory that is allocated to the VM.

20. (Original) The method of claim 19, wherein the execution of the VM is stopped while the VM memory is marked COW.

21. (Original) The method of claim 19, wherein the virtual disk is initially mapped to a parent disk file on a physical disk and wherein the step of maintaining, in an unmodified state, the contents of the virtual disk comprises creating a copy-on-write

(COW) disk file pointing to the parent disk file and mapping the virtual disk to the COW disk file.

22. (Original) The method of claim 19, wherein the step of saving the device state of the VM to the checkpoint data store comprises saving the device state to memory while the VM is not executing and copying the device state from memory to the checkpoint data store after the VM has resumed execution.

23. (Currently amended) A method for generating a checkpoint for a virtual machine (VM) in a virtual computer system, the VM using a virtual disk and a set of VM memory, the method comprising:

maintaining, in an unmodified state, the contents of the virtual disk at the time for which the checkpoint is generated, while allowing the VM to continue ~~using~~ reading from and writing to the virtual disk, the unmodified contents of the virtual disk constituting a checkpointed virtual disk, and the contents of the virtual disk used by the VM constituting an ongoing virtual disk;

saving substantially all of the device state of the VM, including an instruction pointer, a plurality of registers and settings for one or more virtual devices, at the time for which the checkpoint is generated, to a checkpoint data store;

saving the contents of the VM memory, at the time for which the checkpoint is generated, to the checkpoint data store, and allowing the VM to continue ~~using~~ reading from, writing to, and executing code from the VM memory, the contents of the VM memory saved to the checkpoint data store constituting a checkpointed VM memory, and the contents of the VM memory as used by the VM constituting an ongoing VM memory; and

allowing the VM to execute, including executing code from the VM memory, during at least a part of the time during which the checkpoint is being generated, and ensuring that the results of any pending disk writes are applied to both the checkpointed virtual disk and the ongoing virtual disk, that the results of any new disk writes are applied to the ongoing virtual disk, but not to the

checkpointed virtual disk, that the results of any pending disk reads are applied to both the checkpointed VM memory and the ongoing VM memory, and that the results of any new disk reads are applied to the ongoing VM memory, but not to the checkpointed VM memory.

wherein the VM memory is a subset of a physical memory in the virtual computer system, the VM memory being a portion of the physical memory that is allocated to the VM.

24. (Currently amended) The method of claim 23, wherein the step of saving the contents of the VM memory to the checkpoint data store comprises:

marking the set of VM memory as copy-on-write (COW), the set of VM memory constituting original VM memory;

allowing the VM to continue ~~using~~ reading from and writing to the VM memory;

responding to memory COW faults related to the VM memory by generating copies of the original VM memory for read and write use by the VM; and

saving the original VM memory to the checkpoint data store.

25. (Original) The method of claim 24, wherein the step of ensuring that the results of any pending disk reads are applied to the ongoing VM memory comprises reissuing any pending disk reads for which the results of the read were directed to original VM memory for which a COW fault has occurred, but directing the reissued disk reads to the corresponding copies of the original VM memory instead of the original VM memory.

26. (Original) The method of claim 24, wherein the step of ensuring that the results of any new disk reads are not applied to the checkpointed VM memory comprises forcing COW faults for any original VM memory that would otherwise be affected by the new disk reads.

27. (Original) The method of claim 23, wherein the virtual disk is initially mapped to a parent disk file on a physical disk and wherein the step of maintaining, in an unmodified state, the contents of the virtual disk comprises creating a copy-on-write (COW) disk file pointing to the parent disk file and mapping the virtual disk to the COW disk file.

28. (Original) The method of claim 27, wherein the step of ensuring that the results of any pending disk writes are applied to the ongoing virtual disk comprises, if there is a pending disk write to the same COW block group as a subsequent write for which a disk COW fault has occurred, delaying responding to the disk COW fault and delaying the subsequent write until the pending disk write completes.